Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matic Petek

Prototip sistema za vodenje denarnega toka podjetji

diplomsko delo

visokošolski strokovni študijski program
računalništvo in informatika

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Prototip sistema za vodenje denarnega toka podjetji (Prototype of a system for managing company's cash flow)

Tematika naloge:

V okviru diplomske naloge izdelajte prototipa sistema za vodenje prihodkov in odhodkov podjetja (denarnega toka). Aplikacija naj vsebuje spletni vmesnik, kjer bo mogoče uvoziti podatke in opraviti pregled rezultatov. Sistem naj omogoča implementacijo dodatnih algoritmov. V zaključnem delu predstavite ugotovitve in opišite možne nadgradnje razvitega sistema.

Zahvaljujem se mentorju doc. dr. Roku Rupniku za pomoč in svetovanje pri izdelavi diplomskega dela.

Posebno pa bi se zahvalil ženi Barbari Bogdanić Petek za vso izkazano potrpljenje in spodbudo med nastajanjem dela.

Svoji dragi Barbari.

Kazalo

Slike

[]

Seznam uporabljenih kratic

kratica	angleško	slovensko
ERP	Enterprise Resource Planning	Sistem za opravljanje virov
DRY	Do not Repeat Yourself	ne ponavljaj ponovno
HTTP	HyperText Transfer Protocol	protokol na aplikacijski plasti
POJO	Plain Old Java Object	osnovni javanski objekt
API	Application Programming Interface	programski vmesnik
XML	Extensible Markup Language	razširljiv označevalni jezik
JSON	JavaScript Object Notation	format za opis podatkov
HTML	Hyper Text Markup Language	definicija prikaza vsebine
SCRUM	agile software development framework	proces agilnega programiranja
JSNI	JavaScript Native Interface	vmesnik za integracijo kode JavaScript
DOM	Document Object Model	dokumentni objektni model
CSV	Comma-Separated Values	vrednosti ločene z vejico
CSS	Cascading Style Sheets	kaskadna stilska predloga
RPC	Remote Procedure Call	oddaljeni klic čez internetno omrežje

Povzetek

Naslov: Prototip sistema za vodenje denarnega toka podjetji

V diplomskem delu je predstavljen način implementacije prototipa sistema za vodenje prihodkov in odhodkov podjetja (denarnega toka) v obliki spletne aplikacije.

V prvem in drugem poglavju je opisan pomen denarnega toka za spremljanje poslovanja podjetja in omejitve, ki smo jih sprejeli pri implementaciji aplikacije. Sledi tretje poglavje, kjer smo opisali uporabljenje tehnologije in vzroke za izbiro le teh. Navedli smo tudi primere programske kode iz aplikacije, da bi s tem nazorno prikazali argumente za izbiro posamezne tehnologije.

Glavni del diplomske naloge je zajet v četrtem poglavju, kjer na začetku najprej opišemo proces izdelave aplikacije, arhitekture in različne modele. Sledi predstavitev funkcionalnosti aplikacije in na koncu celotni primer uporabe.

V zaključku smo opisali možne nadgradnje obstoječega sistema ter naše ugotovitve.

Ključne besede: poslovanje podjetja, denarni tok, prihodki, Java, GWT.

Abstract

Title: Prototype of a system for managing company's cash flow

The main purpose of the following thesis, is to present the process of developing a prototype of a system that could be used for managing a company's income and outgoings (cash flow) as a web application. In the first and in the second chapter, the importance of a cash flow as a means of monitoring company's finances is demonstrated. It also includes the restrictions that had to be undertaken in the process of implementing the application. The following, third, chapter describes technology used, design patterns and the reasons for their selection. In order to demonstrate the arguments for the presented technology, simple examples are provided to facilitate better understanding. The main part of the thesis is in the fourth chapter, including a presentation of the software development process of the application design, architecture and different design patterns. Furthermore, the application functionality and a use case are presented. Finally, the thesis is rounded off by some ideas for possible future improvements and the thoughts of the authors on the system itself.

Keywords: company finance, cashflow, income, Java, GWT.

Poglavje 1

Uvod

Denarni tok podjetja nam pove, koliko denarja je podjetje prejelo v določenem času in koliko ga je porabilo za poravnavo svojih obveznosti. Natančen pregled nad gibanjem denarnega toka je še posebej pomemben pri novo nastalih podjetjih, kjer običajno lastniki vložijo nekaj zagonskega kapitala, ki pokriva operativno izgubo v začetnem obdobju poslovanja podjetja. Spremljanje denarnega toka je tudi pomembno za že zrela in uspešna podjetja, saj predstavlja enega od kriterijev za spremembe poslovnega modela (večje dolgoročne investicije v nove izdelke in storitve, povečanje števila zaposlenih, spremembe pogojev poslovanja kot so plačilni roki, hitrejše poplačilo kreditov, itd.), ki pa je nujen za stalno prilagajanje spremembam na trgu.

Podatek o gibanju denarnega toka podjetja (tako prihodki kot izdatki) je eden od najbolj verodostojnih pokazateljev, kako uspešno posluje podjetje. Prikaže namreč, kako uspešno se podjetje prilagaja vsem izzivom, ki jih ima na trgu - na eni strani prodaja izdelkov in izterjave denarja ter na drugi strani obvladovanje svojih lastnih stroškov.

Posebno vlogo pa igra denarni tok tudi pri novo nastalih podjetjih, kjer lastniki običajno vložijo zagonski kapital, ki se potem porablja za lansiranje izdelka na trg. Spremljanje mesečne porabe denarja glede na prihodke (ang. *burn rate*) je eden od ključnih pokazateljev, koliko časa (glede na denar) ima še podjetje, da pripravi ustrezen produkt, ki mu bo prinašal prihodke. Na

podlagi te informacije lahko vodstvo lažje sprejema odločitve o dodajanju in omejevanu funkcionalnosti izdelka. Cilj diplomska naloge je izdelava aplikacije, ki omogoča zajem prihodkov in odhodkov podjetja ter spremljanje gibanja denarnega toka. Uporabnik lahko na enostaven način pripravi vhodne podatke iz obstoječega informacijskega sistema ter jih uvozi v aplikacijo. Na podlagi preteklih podatkov lahko spremlja napoved gibanja prihodkov in odhodkov v prihodnosti glede na stanje gotovinskega salda. Uporabniški vmesnik omogoča pregled podatkov po različnih časovnih komponentah (dan, teden, mesec). Vodilo enostavne uporabe smo uporabili tudi pri izbiri načina napovedovanja prihodkov na podlagi preteklih vhodnih in izhodnih računov. Menimo, da z uporabo enostavnih algoritmov lahko dosežemo dovolj veliko stopnjo zanesliivosti in uporabnikom omogočamo sledljivost informacij do izvornega podatka.

V diplomski nalogi smo tudi morali postaviti ločnico med napovedjo denarnega toka in napovedjo prodaje. Pri denarnem toku izhajamo izključno iz dokumentov (vhodni in izhodni računi), ki so že nastali in imajo veliko verjetno realizacije. Delno lahko uporabnik doda dodatne informacije (recimo dogovorjen rok plačila). Periodična izdaja sicer sodi med ocene v prihodnosti, vendar smo se odločili, da se ta mehanizem lahko uporabi, ker prenaša veliko verjetnost za realizacijo (npr. mesečna najemnina, plačilo elektrike, itd.). Zato se je potrebno zavedati, da je pogled v prihodnost s tem zelo omejen in mora uporabnik na podlagi svojih izkušenj iz načina poslovanja oceniti kako daleč v prihodnost je napoved še relevantna. Pri napovedi prodaje pa si zastavimo več dogodkov v prihodnosti, za katere ocenimo predvidene stroške in prihodke z upoštevanjem časovne komponente. Dogodke potem postavimo na časovno os in poizkušamo ugotoviti ali lahko iz obstoječih virov (finančni, človeški) izvedemo realizacijo.

Poglavje 2

Denarni tok

Izraz denarni tok nam pove, koliko denarja je priteklo na transakcijski račun podjetja (prejemki) ter s kakšnimi transakcijami in koliko denarja je odteklo z računa podjetja (izdatki) v določenem časovnem obdobju [1].

Izraz lahko definiramo tudi kot "temeljni računovodski izkaz, v katerem so resnično in pošteno prikazane spremembe stanja denarnih sredstev in njihovih ustreznikov za poslovno leto ali medletna obdobja, za katere se sestavlja. Spada med izvedene računovodske izkaze in izkazuje denarni tok med dvema obdobnima bilancama stanja, ki jih je podjetje v posameznem obdobju ustvarilo pri poslovanju pri ustvarjanju proizvodov in storitev, naložbah oziroma nalaganju finančnih sredstev v investicije in finančne naložbe, ter financiranju ali pridobivanju finančnih sredstev iz zunanjih virov in njihovem vračanju. Na podlagi denarnih tokov posameznih dejavnosti lahko sklepamo iz katerih virov podjetje pridobiva denarna sredstva in kje jih porablja.". [2]

V uvodu smo omenili, da je pogled v prihodnost zelo omejen, saj izhajamo izključno iz nastalih dokumentov. Poglejmo si primer uporabe na dveh podjetjih. Podjetje A ima zelo kratke plačilne roke (npr. 30 dni), nima periodično izdanih računov in na mesec izda 30 računov po enako razporejenem mesečnem časovnem obdobju. Stroške ima relativno konstantne (plače, najemnine). Na podlagi dokumentov lahko sklepamo, da bo na prihodkovni strani pogled relevanten samo za naslednjih 30 dni, ker ostali dokumenti še

niso izdani. Na drugem primeru - Podjetje B - pa imamo nekoliko drugačni scenarij. Večina računov (80

Natančen izkaz denarnih tokov in predvidevanja o višini in dinamiki denarnih tokov v prihodnosti je pogoj za ohranjanje plačilne sposobnosti podjetja, zato je zanimiv tudi za banko kot posojilodajalca. Banka tudi želi ugotoviti glavne vire prejemkov in izdatkov v preteklem obdobju. Zelo pomembni so tudi viri, saj se lahko zgodi, da iz nekaterih virov ne bo prejemkov v prihodnosti. Banko zanima, če lahko denarni tok iz poslovanja pokriva vse finančne obveznosti, razne investicije in morebitne rezerve, ki bi zagotavljale plačilno sposobnost podjetja v prihodnosti. S primerjavo denarnih tokov v zadnjih letih lahko banka ugotovi povezanost med denarnim tokom iz poslovanja oziroma financiranja ter tako ugotovi odvisnost podjetja od zunanjih virov. [3]

Podjetje ima lahko dobiček, a je njegov obstoj kljub temu ogrožen. Stanje denarja na transakcijskem računu je lahko eden glavnih »izdajalcev« kreativnega računovodje. Dobiček se lahko tudi umetno ustvari, sredstva se lahko prevrednotijo, zaloge napihnejo, terjatve ne odpisujejo, računovodski izkazi se lahko priredijo celo za nazaj. [4] Stanje denarja pa je absolutna številka, ki se je ne da prirejati.

Poglavje 3

Uporabljena orodja in tehnologije

Aplikacijo sestavlja spletni vmesnik, zaledna aplikacija in shranjevanje podatkov v bazo. Spletni vmesnik je izdelan s pomočjo tehnologije GWT, ki omogoča razvoj spletnih aplikacij v programske jeziku Java. Ključni podatki so prikazani s pomočjo grafov, za katere se uporablja odprtokodna rešitev FusionCharts. Z zaledno aplikacijo komunicira preko protokola GWT-RPC, ki je optimizirana oblika RPC protokola nad HTTP protokolom, kjer se ravno tako uporablja Java programski jezik. Preračunavanje datumskih območji se izvaja s pomočjo zelo razširjene in preizkušene Java knjižnice JodaTime. Apache POI omogoča enostavno branje Excel datotek.

3.1 GWT

Eden večjih izzivov pri pisanju več nivojskih aplikacij je uporaba skupne poslovne logike med različnimi nivoji. Prednosti skupne programske logike so naslednji :

- implementacija poslovnega programa se nahaja samo na enem mestu v programski kodi,
- vsaka sprememba je takoj dostopna na različnih nivojih,

- sistemi za zagotavljanje kakovosti (npr. avtomatični testi, analiza pokritosti programske kode, itd) se vedno izvajajo samo na enem delu kode.

S skupno programsko logiko tudi najlažje sledimo zelo priporočljivem vzorcu v programiranju - DRY (ang. *Do not repeat yourself*) - ki nam na dolgi rok vedno pripomore k večji preglednosti programske kode in lažjemu vzdrževanju aplikacije. V praksi se zato velikokrat uporabljajo prevajalniki ali ogrodja, ki omogočajo transformacijo želenega programskega jezika v domorodni programski jezik določene platforme. V kolikor želimo Java programski jezik uporabiti tudi znotraj spletnih brskalnikov brez uporabe vtičnikov, je potrebno programsko kodo transformirati v JavaScript programski jezik. Že več kot 10 let je najbolj priljubljeno orodja GWT, ki omogoča :

- pisanje programske kode v Javi in transformacijo v JavaScript,
- enostaven dostop do JavaScript vmesnikov preko tehnologije JSNI oz. JSinteropr [6],
- komunikacijo z Java zalednim sistemom preko tehnologije GWT-RPC [7].

3.2 GQuery

Ogrodje GWT vsebuje tehnologijo GWT Widgets, ki omogoča delo s spletnimi gradniki v Java kodi. Programerju preko abstrakcije zakrije delovanje HTML, CSS in JavaScript kode v spletnem brskalniku, tako da lahko spletne gradnike programiramo kot običajne Java razred z uporabo opazovalec programskega vzorca (ang. *observer design pattern*).

```
Button b = new Button("Zagon");
b.addClickHandler(new ClickHandler() {
   public void onClick(ClickEvent event) {
          Window.alert("Pritisnil sem gumb");
   }
});
```

Slika 3.1: Uporaba GWT Widget za izdelavo HTML gumba in prestrezanje dogodka ob kliku

Na žalost ima tehnologija GWT Widget tudi veliko slabost, da je model dogodkov prilagojen času, ko so se osnovne funkcionalnosti v brskalniku še zelo razlikovale med seboj (čas Internet Explorer 6.0) in so imeli zelo slabo podporo za izvajanje programskega jezika JavaScript. Izgradnja bolj zahtevnih komponent zahteva združevanje osnovnih komponent, kar posledično pomeni veliko dodatne HTML ter CSS kode in tako postanejo kompleksne komponente težavne za oblikovanje in počasne za delovanje.

Kot možna alternativa je izgradnja komponent v čisti HTML in CSS kodi in dodajanje dogodkov preko knjižnice GQuery. Programski vmesnik (ang. *API*) GQuery je prilagojen izredno razširjeni JavaScript knjižnici jQuery, kar omogoča hitro učenje preko primerov, ki jih najdemo na internetu.

```
<input type="button" class="MKTimeline_startButton" value="Danes"/>

$(".MKTimeline_startButton").click(new Function(){
   @Override
   public void f() {
      $("Sporocilo").insertAfter("#MKTimeline_row");
   }
});
```

Slika 3.2: Primer gumba z dogodkom v GQuery

Zgoraj opisan pristop (glej sliko 3.2) smo tudi uporabili za aplikacijo diplomske naloge, saj smo morali zgraditi relativno zahtevno komponento za prikaz in vnos podatkov, z možnostjo bolj podrobnega vpogleda v podatke.

3.3 HTML, CSS in JavaScript

HTML (angl. HyperText Markup Language) je standard, ki definira strukturo spletnih strani. Prve uradne specifikacije so bile objavljene leta 1993 [9], jezik pa je spisan s pomočjo značk (angl. *tags*), ki so obdani s oglatimi oklepaji. Danes se uporablja predvsem za prikaz vsebine spletnih strani ter osnovno strukturo spletnih aplikacij. Podobno spletnih strani določimo s pomočjo jezika CSS (angl. *Cascading Style Sheets*), prva verzija standarda pa je bila objavljena leta 1996 [10]. Osnovno vodilo pri izdelavi programskega jezika je bila želja po ločitvi vsebine dokument (HTML) od njegove prezentacije (CSS) z vidika načrta vsebine, barv ter pisav. Ločevanje v naslednjih fazah omogoča enostavnejše prilagajanje in več kontrole na različnih sistemih prikazovanja (spletni brskalnik, mobilni telefon, tiskanje vsebine, itd), še posebej če se spreminjajo razmerja velikost naprave (npr. obračanje telefona). Tako lahko isto spletno HTML stran učinkovito prikažemo na različnih napravah, kar zmanjšuje kompleksnost in potrebo po več kopijah vsebine. Zadnja verzija CSS 3 je razbita na mnogo modulov, ki se razvijajo neodvisno od glavne verzije.

Dinamičnost spletnih aplikacij dosežemo s pomočjo skriptnega programskega jezika JavaScript, ki je bil primarno razvit za potrebe DOM manipulacije HTML elementov. V preteklosti se je uporabljal predvsem znotraj spletnih brskalnikov, zadnje časa pa ga lahko uporabimo tudi za razvoj zalednih sistemov. Z leti je dobil vse sintaktične funkcionalnosti sodobnih programskih jezikov, od Jave pa se loči predvsem po tem, da je dinamičen in strogo ne zahteva določitev tipa spremeljivk. To se je izkazalo kot pomanjkljivost predvsem pri bolj obsežnih aplikacijah, zato so nastale izpeljanke, ki predvsem podpirajo strukturirane podatkovne tipe ter preverjanje preverja-

nje kode (npr. odstranjujejo t.i. mrtvo kodo). Najbolj znane so TypeScript in Closure.

V diplomski nalogi JavaScript nismo uporabili neposredno, saj je celotni spletni del aplikacije napisan u uporabo orodja GWT, ki prevede Java v JavaScript. Tudi jezik HTML smo uporabili v zelo omejeni funkcionalnosti, ker v naši spletni aplikaciji ni bila potreba po predstavitvi večjih vsebin na različne načine. Komponente za prikaz vsebine se namreč najbolj optimalno gradi z uporabo navideznih gradnikov *div*, ki predstavljajo prostor za ostale gradnike. Vsebino le teh pa potem dinamično določimo z dinamičnih jezikom in oblikujemo podobo preko CSS.

```
▼<td align="left" style="vertical-align: top;">
  ▼<div>
    ▶<div class="MKTimeline_header">…</div>
    ▶<div class="MKTimeline_chart" id="MKTimeline_chart">…</div>
    ▶<div class="MKTimeline_title" id="MKTimeline_title">…</div>
    ▼<div id="MKTimeline_row_list">
      ▶<div id="MKTimeline_row_0" class="MKTimeline_row">…</div>
      ▼<div id="MKTimeline_row_1" class="MKTimeline_row">
          <div class="MKTimeline_row_firstcell_row">Stanje bančni račun</div>
        ▼<div id="MKTimeline_row_cell_1_0" class="MKTimeline_row_cell">
            <a onclick="__mkTimeLineOpenDetail(1,0)" href="javascript:void(0);">1200</a>
          </div>
          <div id="MKTimeline_row_cell_1_1" class="MKTimeline_row_cell"> </div>
          <div id="MKTimeline_row_cell_1_2" class="MKTimeline_row_cell"> </div>
          <div id="MKTimeline_row_cell_1_3" class="MKTimeline_row_cell"> </div>
          <div id="MKTimeline_row_cell_1_4" class="MKTimeline_row_cell"> </div>
          <div id="MKTimeline_row_cell_1_5" class="MKTimeline_row_cell"> </div>
          <div id="MKTimeline_row_cell_1_6" class="MKTimeline_row_cell"> </div>
        </div>
      ▶<div id="MKTimeline_row_2" class="MKTimeline_row">…</div>
      ▶<div id="MKTimeline_row_3" class="MKTimeline_row">…</div>
      ▶<div id="MKTimeline_row_4" class="MKTimeline_row">…</div>
      ▶<div id="MKTimeline_row_5" class="MKTimeline_row">…</div>
      ▶<div id="MKTimeline_row_6" class="MKTimeline_row">…</div>
      ▶<div id="MKTimeline_row_7" class="MKTimeline_row">…</div>
      ▶<div id="MKTimeline_row_8" class="MKTimeline_row">…</div>
      ▶<div id="MKTimeline_row_9" class="MKTimeline_row">…</div>
      ▶<div id="MKTimeline_row_10" class="MKTimeline_row">…</div>
      ▶<div id="MKTimeline_row_11" class="MKTimeline_row">…</div>
```

Slika 3.3: HTML struktura tabele za prikaz podatkov

3.4 MySQL

MySQL podatkovna zbirka je postala priljubljena predvsem ob hitrem širjenju spletnih strani in spletnih aplikacij, kjer se je pokazala potreba po enostavnih podatkovnih zbirkah, ki lahko delujejo tudi na manj zmogljivi strojni opremi. Z leti je pridobila veliko funkcionalnosti večjih podatkovnih zbirk (predvsem podporo za transakcije in replikacije) in še vedno ohranila možnost enostavne uporabe in hitrega delovanja.

V diplomski nalogi uporabljamo samo osnovno funkcionalnost trajnega shranjevanja podatkov v tabele in branja pred izgradnjo modela podatkov v Java objektih.

3.5 Apache Joda-Time

Preračunavanje med datumi v programske jeziku Java ni bilo ustrezno podprto vse do sedanje verzije Java 8. V mislih imamo predvsem izračune tipa *kakšno je datumsko območje naslednjih N tedno* ali *datum čez 60 dni, kjer se upošteva dinamičnost dni v mesecu.*

Pred leti se je uveljavila knjižnica Joda-Time [10], ki smo jo tudi zaradi izkušenj uporabili za izdelek diplomske naloge. Uporabljamo jo predvsem za izračun predvidenega datuma novega plačila računa na podlagi rezultata algoritma in določitev tedenskega območja za prikaz podatkov na grafu.

```
DateTime dt = new DateTime();
DateTime last = dt.dayOfMonth().withMaximumValue();
```

Slika 3.4: Zadnji dan tekočega meseca

3.6 Apache POI

Aplikacija omogoča uvoz obstoječih podatkov preko Excel datoteke, zato smo potrebovali knjižnico, ki nam omogoča enostavno branje Excel formata

znotraj Java programskega okolja. Apache POI [11] je že vrsto let ena od najbolj priljubljenih rešitev, ki med drugim podpira tudi ostale Microsoft formate (npr. Word). V diplomski nalogi smo uporabili samo del, ki omogoča branje Excel datotek, kot je prikazano na spodnjem primeru.

```
public class DataImporterProductFormatExcel extends
    DataImporterProductFormat {
   protected int xlsRowIndex = 0;
   protected Sheet xlsSheet = null;
   protected Row xlsRow = null;
   protected DBTools dbTools = new DBTools();
   public DataImporterProductFormatExcel(DataImporterSettings _s) {
      super(_s);
   }
   public void open(byte[] _fileContent) throws Exception {
      xlsSheet =
          DataImportUtils.getInstance().getFirstSheetFromFile(_fileContent);
      FormulaEvaluator evaluator =
          xlsSheet.getWorkbook().getCreationHelper().createFormulaEvaluator();
      xlsRowIndex = xlsSheet.getFirstRowNum();
   }
   public boolean nextRecord() throws Exception {
      int emptyRowCount = 0;
      for (;emptyRowCount < 5; emptyRowCount++) {
         xlsRow = xlsSheet.getRow(xlsRowIndex);
         xlsRowIndex++;
         if (!DataImportUtils.getInstance().isRowEmpty(xlsRow)) {
            break;
         }
      }
      if (emptyRowCount < 5) { return true;
      } else { xlsRow = null; return false; }
   }
   public int columnCount() throws Exception {
      return xlsRow.getLastCellNum();
   }
   public String getString(int _i) throws Exception {
      return dbTools.parseForceStringFromCell(xlsRow.getCell(_i),
          true);
   }
}
```

Slika 3.5: Ovoj (ang. *wrapper*) okoli knjižnice Apache POI za branje Excel vrstic in posameznih celic

3.7 FusionCharts

V aplikaciji diplomske naloge vizualizacijo predstavimo z dvema grafičnima elementoma : grafom prihodkov / odhodkov in gibanja denarja ter tabelo za natančnejši vpogled v podatke. Za graf smo uporabili razširjeno knjižnico FusionCharts [12], ki omogoča predstavitev podatkov v veliko različnih oblikah grafov. V preteklosti so bili grafi narejeni s pomočjo tehnologije Adobe Flash, danes pa se praktično uporabljajo samo še HTML5 grafi. Knjižnico smo izbrali predvsem zaradi uporabe na preteklih projektih. Za izdelavo grafa moramo nastaviti :

- tip grafa,
- opisni podatki,
- podatki za vizualizacijo, ki so lahko v JSON ali XML obliki.

```
{
   "type":"mscolumnline3d",
   "renderAt":"chartContainer",
   "width":"80%",
   "height":"300",
   "dataFormat":"json",
   "showLegend":"0",
   "dataSource":{
      "chart":{ "showvalues":"0", "yaxisvaluespadding":"10",
         "numberprefix":"", "showborder":"0" },
      "categories":[ { "category":[ { "label":"06.2016" } ] } ],
      "dataset":[ { "seriesname":"Skupaj prilivi", "data":[ {
         "value":"8000" } ] } ]
   }
}

public static native void showChart(JavaScriptObject jsdata) /*-{
        $wnd.FusionCharts.ready(function(){
        var revenueChart = new $wnd.FusionCharts(jsdata);
        revenueChart.render("MKTimeline_chart_content");
   })
}-*/;
```

Slika 3.6: Podatki za opis grafa v JSON obliki ter klic knjižnice FusionChart, ki generira graf

Poglavje 4

Aplikacija

4.1 Analiza in načrt

Razvoj aplikacije smo izvajali po t.i. SCRUM metodologiji [13].Za začetek smo določili glavne zgodbe (ang. *stories*) naše aplikacije :

- uporabniška izkušnja (na kakšen način bodo uporabniki izvedli prenos preteklih podatkov ter pregledovali napovedi plačil),
- shranjevanje podatkov (način shranjevanja preteklih podatkov in ročno vnešenih dodatnih podatkov ter ustrezne POJO preslikave),
- izračun napovedi plačil (nič izvedbe napovedi plačil ter opisa izvora napovedi),
- prenos podatkov iz obstoječega sistema (način prenosa obstoječih podatkov v model za izračun napovedi),
- spletni del aplikacije (komunikacijski protokol med spletno aplikacijo in zaledno aplikacijo ter zagon spletne aplikacije).

Po določitvi glavnih zgodb, smo posamezne razdelili na podzgodbe in jim ocenili zahtevnost. Naslednji korak po SCRUM metodologiji je določitev zahtevnosti na podlagi točkovnega sistema (ang. *story points*). Podobno

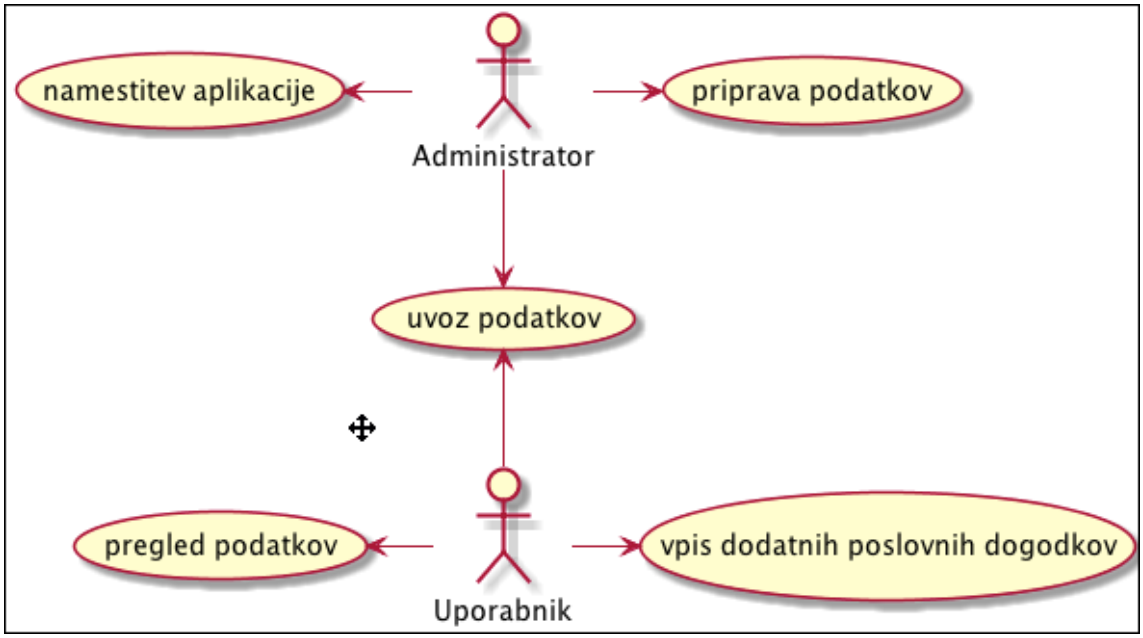
kot predvideva metodologija, smo tudi mi določili točke na podlagi tehnične zahtevnosti in ocene zahtevanega časa. Naloge smo zatem vpisali v dnevnik nalog (ang. *backlog*). Po SCRUM metodologiji bi sicer morali določiti časovne okvirne (ang. *sprint*), ki običajno trajajo 14 dni. Vendar smo se zaradi koordinacije dela z ostalimi obveznostmi (družina, služba) raje odločili za kombinacijo s Kanban metodologijo [14]. Le ta predvideva, da vedno na podlagi backlog določimo ključno funkcionalnost, ki jo potem razvijemo do konca. Za izvedbo uporabimo toliko časa, da funkcionalnost dokončamo v zahtevani kvaliteti.

4.1.1 Diagram primera uporabe

Za uporabo aplikacije smo predvideli naslednje akterje (ang. *actor*) :

- sistemski administrator - običajno se podatki o poslovanju nahajajo v obstoječem PIS (poslovnem informacijskem sistemi - ang. *ERP*). Potrebno jih je izvoziti v Excel obliko, ki je primerna za uvoz v aplikacijo denarnega toka. V primeru, da podatke ni mogoče enostavno izvoziti in/ali so potrebne še dodatne obdelave, običajno potrebujemo osebo s poznavanjem delovanja informacijskih sistemov.

- uporabnik aplikacije - najpomembnejša vloga je pregled podatkov napovedi denarnega toka ter interpretacija informacij na podlagi pridobljenih podatkov. Predpostavlja se, da ima tudi domensko znanje o poslovanju podjetja.



Slika 4.1: Akterji in njihova uporaba aplikacije

Predvideni so naslednji primeri uporabe aplikacije (ang. *use case*) :

- namestitev aplikacije - vzpostavitev delovanja aplikacije na osebnem ali strežnik računalniku,
- priprava podatkov - izvoz podatkov iz obstoječega PIS ter uvoz v denarni tok aplikacijo,
- uvoz podatkov - izvedba uvoza podatkov in odprava morebitnih napak,
- vpis dodatnih poslovnih dogodkov - uporabnik aplikacije lahko vnešene podatke dopolni z dodatnimi oznakami za poslovne dogodke (npr. vpis stanja na bančnem računu, določitev datuma plačila računa, račun je neizterljiv),
- pregled podatkov - uporabnik aplikacije se lahko sprehaja po časovni komponenti podatkov (pregled po dnevih / tednih / mesecih v prihodnost) ali izvaja podrobnejši vpogled v vsebino podatkov.

4.1.2 Podatkovni model

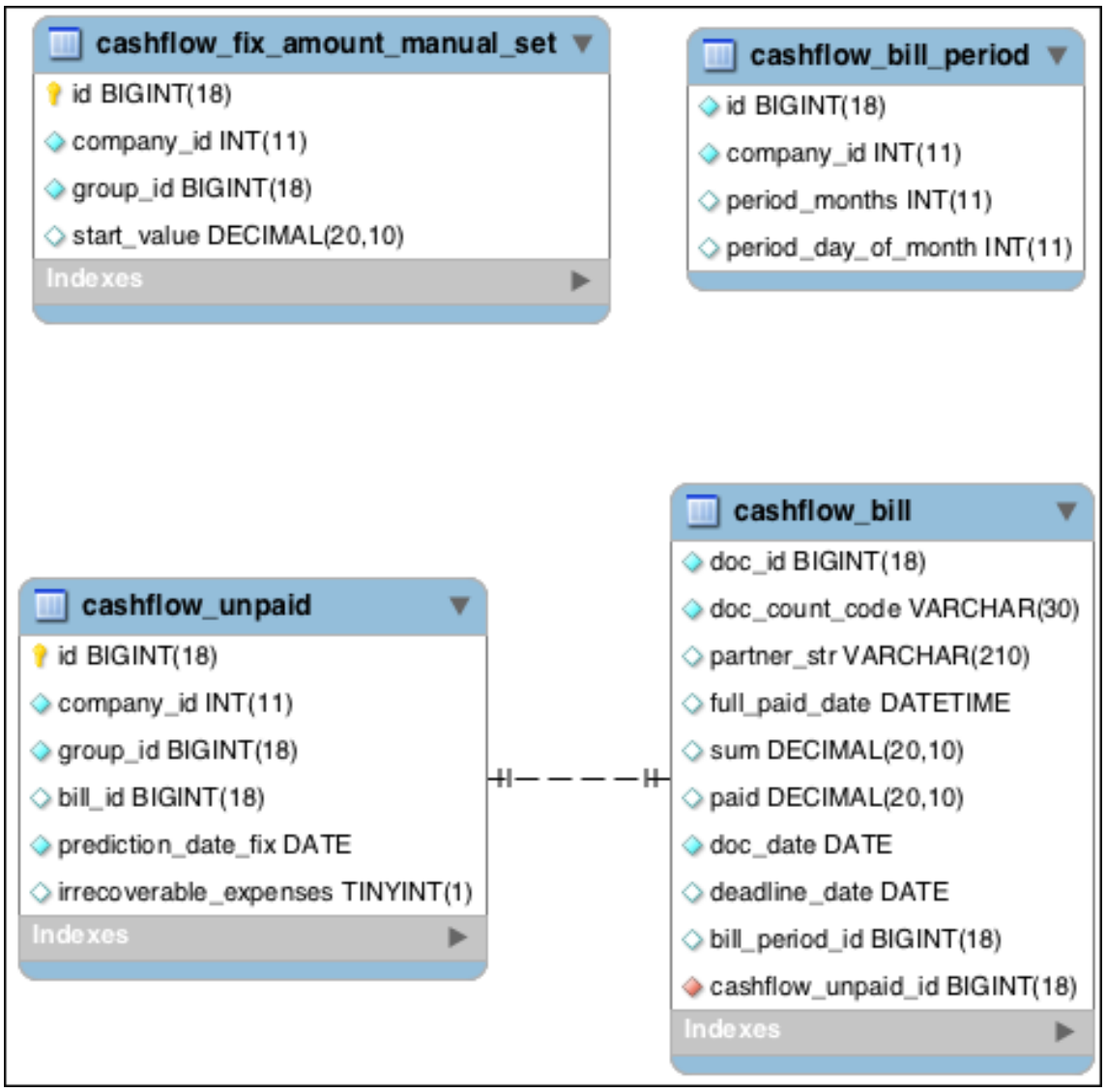
Vsi podatki začetnega uvoza in naknadnih sprememb so shranjeni v MySQL podatkovni bazi. Pri tem se uporabljajo naslednje tabele :

- `cashflow_bill` - glavna tabela, v katero shranjujemo podatke o računih ; lahko vsebuje povezavo (`bill_period_id`) na tabelo z opisom periode ponavljanja,

- `cashflow_bill_period` - v primeru, da ima račun periodiko, je v tabeli zapisan podatek o število mesecev ponavljanja (`period_months`) ter `period_day_of_month` (kateri dan v mesecu se izvede ponavljanje),

- `cashflow_fix_amount_manual_set` - ročno nastavljanje vrednosti (`start_value`) za posamezne plačilne inštrumente (`group_id`),

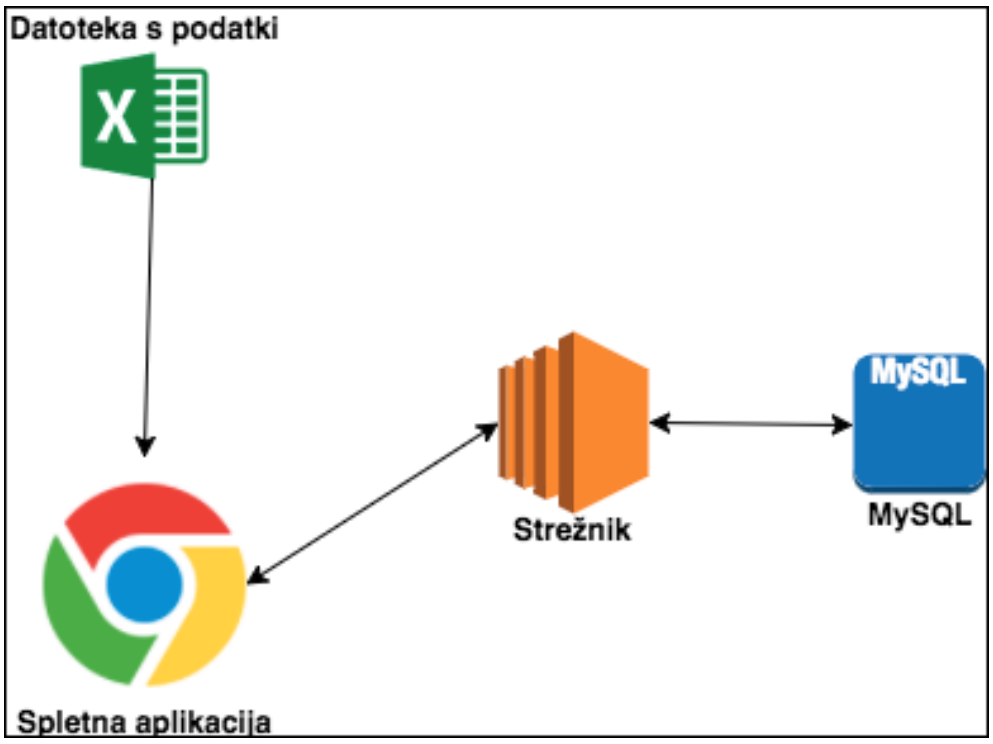
- `cashflow_unpaid` - dodatni podatki za neplačane račune: ročni vnos predvidenega datuma plačila (`prediction_date_fix`) in oznaka, da račun ne bo nikoli plačan (`irrecoverable_expenses`).



Slika 4.2: Podatkovni model za shranjevanje podatkov

4.2 Arhitektura

Za lažjo predstavo o celotnem sistemu smo pripravili skico najpomembnejših arhitekturnih sklopov.



Slika 4.3: Arhitektura aplikacije

Datoteka s podatki nam služi za uvoz podatkov o prilivih in odlivih, ki nam potem služijo za izdelavo napovedi denarnega toka. Uporabnik vso interakcijo z aplikacijo izvaja preko spletne aplikacije. Aplikacija je primarno namenjena zajemu, prikazu in navigaciji po podatkih. Komunicira s strežnikom, kjer se nahaja zaledna aplikacija. Le ta upravlja s podatki - sprejema nove podatke od spletne aplikacije, izvaja izračune, trajno shranjuje podatke v bazo podatkov ter pripravlja podatke za prikaz v spletni aplikaciji. Baza podatkov je namenjena trajnemu shranjevanju uvoženih podatkov in sprememb, ki jih uporabnik izvede preko spletne aplikacije.

4.2.1 Priprava podatkov za uvoz

Začetne podatke v sistem uvozimo preko nabora podatkov (ang. *dataset*) v Excel obliki (glej slika 4.4).

Področje	Tip	Številka računa	Partner	Neto znesek	Datum	Rok plačila	Datum plačila
Priliv	Domači	D101	Mercator d.d.	3000	25.5.2016	2.6.2016	
Priliv	Domači	D102	Krka d.d.	1500	26.5.2016	3.6.2016	
Priliv	Domači	D103	Sava d.d.	4000	1.5.2016	15.5.2016	
Odliv	Domači	P100	Poslovni prostori d.o.o	800	27.5.2016	5.6.2016	
Odliv	Domači	Plače	Moje podjetje	3500	21.6.2016	6.6.2016	

Slika 4.4: Primer podatkov v Excel obliki

Polja so razdeljena v dve skupine: obvezni podatke in pomožni podatki, ki jih lahko naknadno vnesemo tudi preko spletne aplikacije. Obvezni podatki so naslednji :

- `področje` - izhodne račune označimo kot “Priliv”, vhodne pa kot “Odliv”,
- `tip` - glede na drevesno strukturo razdelitve tipov računov določimo, v katero skupino se bo račun preslikal,
- `številka računa` - oznaka po kateri lažje identificiramo račun v drugem informacijskem sistemu ali fizični obliki,
- `partner` - poljubni opis partnerja. Podobno kot številka računa nam pomaga pri identifikaciji računa,
- `znesek` - znesek računa, ki ga partner ali naše podjetje mora plačati,
- `datum` - datum izdaje računa,
- `rok plačila` - datum zapadlosti računa oz. datum, do katerega se pričakuje, da bo račun plačan,
- `datum plačila` - v kolikor je bil račun že plačan, je to datum, ko je bilo dejansko izvedeno polno plačilo računa.

Pomožni podatki predstavljajo dodatna stanja računov :

- `obdobje - št. mesecev` - v kolikor imamo ponavljajočo izdajo (npr. mesečna naročnina) ali prejem (npr. elektrika, voda, najemnina poslovnih prostorov, telekomunikacijske storitve, zavarovanja, plače, itd)

računov, lahko določimo mesečno obdobje, za katerega se bodo računi upoštevali tudi v prihodnosti. Npr. račun za najem poslovnih prostorov običajno prejmemo vsak mesec, medtem ko zavarovanje plačujemo na 12 mesecev,

- `obdobje - dan v mesecu` - v kolikor imamo vključeno ponavljajočo izdajo računov moramo bolj natančno določiti tudi dan v mesecu, ko bo račun izdan oz. predvidoma prejet,
- `ocena plačila` - v kolikor račun še ni plačan in je rok plačila že potekel, lahko podamo oceno predvidenega prejema plačila. V praksi jo običajno določimo na podlagi predhodnega dogovora s stranko,
- `neizterljivo` - v kolikor že vemo, da račun ne bo plačan, ga lahko označimo kot račun, ki ga ne bomo mogli izterjati. Posledično se le tak račun tudi ne bo upošteval pri napovedi denarnega toka.

4.2.2 Implementacija algoritma za napoved

V program lahko zelo enostavno dodamo novo implementacijo algoritma za napoved plačila računov. Potrebno je narediti novi razred, implementirati vmesnik `CashFlowPredictionInterface` (glej slika 4.5). ter vključiti implementacijo v nastavitve zagona aplikacije (glej 4.3.1. Zagon aplikacije).

```
public interface CashFlowPredictionInterface {
   public String getId();
   public void predict(List<CashFlowDocumentStruct> _billList, Date
       _fromDate) throws Exception;
}
```

Slika 4.5: Vmesnik za implementacijo algoritmov

Metoda `getId` vrne edinstveno oznako implementacije vmesnika (lahko je kar pot in ime razreda) in se lahko uporabi na uporabniškem vmesniku

aplikacije za izbiro tipa algoritma. Dejansko implementacijo algoritma pa izvedemo v metodi `predict`. Pri tem dobimo kot vhod seznam vseh računov (glej slika 4.5) in današnji datum. Le tam nam služi za lažjo orientacijo, kaj napovedujemo za prihodnost in kaj za preteklost. Metoda mora ob svojem izhodu na vseh strukturah tipa `CashFlowDocumentStruct` določiti naslednje parametre :

- `prediction_date` - datum, kdaj naj bi bil račun predvidoma plačan,
- `prediction_desc` - pojasnilo za uporabnika o načinu določitve datuma plačila računa.

Na sliki 4.6 je predstavljen zelo enostaven primer algoritma, kjer za napoved plačila kar vedno uporabimo rok plačila računa.

V metodi `predict` moramo upoštevati tudi naslednja stanja :

- račun ima oznako Neizterljivo,
- uporabnik je preko uvoza podatkov ali spletne maske določil oceno prejema denarja, račun je že bil plačan (običajno velja za pretekle račune).

```
public class CashFlowPredictionDeadLine implements
    CashFlowPredictionInterface {
   StructTools st = StructTools.getInstance();

   @Override
   public String getId() {
      return "deadline";
   }

   @Override
   public void predict(List<CashFlowDocumentStruct> _billList, Date
       _fromDate) throws Exception {
      for (CashFlowDocumentStruct cfds : _billList) {
         if (st.isTrue(cfds.getIrrecoverable_expenses())) {
            cfds.setPrediction_desc("Ne bo placljiv");
         } else if (cfds.getPrediction_date_fix() != null) {
            cfds.setPrediction_desc("Tocno vnesen datum");
         } else if (cfds.getFull_paid_date() != null) {
            cfds.setPrediction_date(cfds.getFull_paid_date());
            cfds.setPrediction_desc("Ze placan preko uvoza
                podatkov");
         } else {
            cfds.setPrediction_date(cfds.getDeadline_date());
            cfds.setPrediction_desc("Enako kot je datum placila");
         }
      }
   }
}
```

Slika 4.6: Enostaven primer implementacije algoritma

4.3 Predstavitev uporabe aplikacije

4.3.1 Zagon aplikacije

Aplikacije deluje v standardnem okolju za Java spletne aplikacije. Za zagon potrebujemo strežnik z dodatnimi programi :

- zagonsko okolje Oracle Java 7 ali več,
- spletni strežnik Tomcat 7 ali več,
- podatkovna baza MySQL 5.6 ali več.

Po ustrezni namestitivi aplikacije (postopek je opisan v datoteki readme.txt), je potrebno nastaviti spodnje parametre, ki se nahajajo v datoteki cashflow.properties (glej 4.7):

- `db_connection_string` - opis povezave do MySQL podatkovne baze,
- `prediction_class_list` - seznam razredov, ki vsebuje implementacije algoritma za napoved.

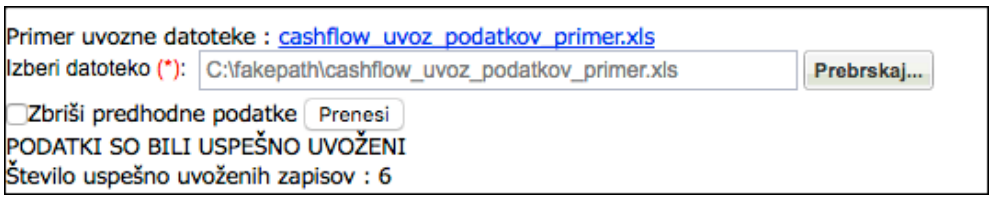
```
db_connection_string=jdbc:mysql://localhost:3307/cashflowdb?user=test&password=mydb123
prediction_class_list=si.cashflow.CashFlowPredictionLastBills,si.cashflow.CashFlowPredictionLastBills
```

Slika 4.7: Zahtevani parametri za zagon aplikacije

4.3.2 Uvoz podatkov preko Excel

Za dobro napoved potrebujem čim več podatkov o že plačanih računih. Ker je večini uporabnikov Excel oblika podatkov dobro poznana in ker obstoječi informacijski sistemi večinoma podpirajo izvoz podatkov v Excel ali CSV obliko, smo naredili podporo za uvoz podatkov preko prilagojene Excel tabele. Kot prikazuje slika 4.8, v aplikaciji izberemo datoteko in podatki se bodo uvozili. V primeru napak dobimo ustrezne informacije, na katerih mestih se le te nahajajo. Vsak naslednji uvoz zbriše podatke računov, vendar ohrani pomožne podatke računov (obdobje ponavljajoče izdaje, ocena plačila,

neizterljivo). Na ta način lahko urejamo podatke v Excel datoteki in jih ponavljajoče uvažamo, brez da bi bilo potrebno ponovno nastavljati pomožne podatke. V kolikor pa bi radi izvedli čisti uvoz podatkov, lahko vključimo možnost "Zbriši tudi pomožne podatke".



Slika 4.8: Izbira datoteke s podatki in uspešno izveden uvoz

4.3.3 Ročni vnos stanja na plačilnih inštrumentih

Podjetje za svoje poslovanje uporabljajo različne plačilne inštrumente, kot so :

- transakcijski računi,
- blagajne, kjer se jim vedno del denarja nahaja kot menjalnina,
- spletni plačilni inštrumenti, kot so PayPal, Telekom Moneta, itd

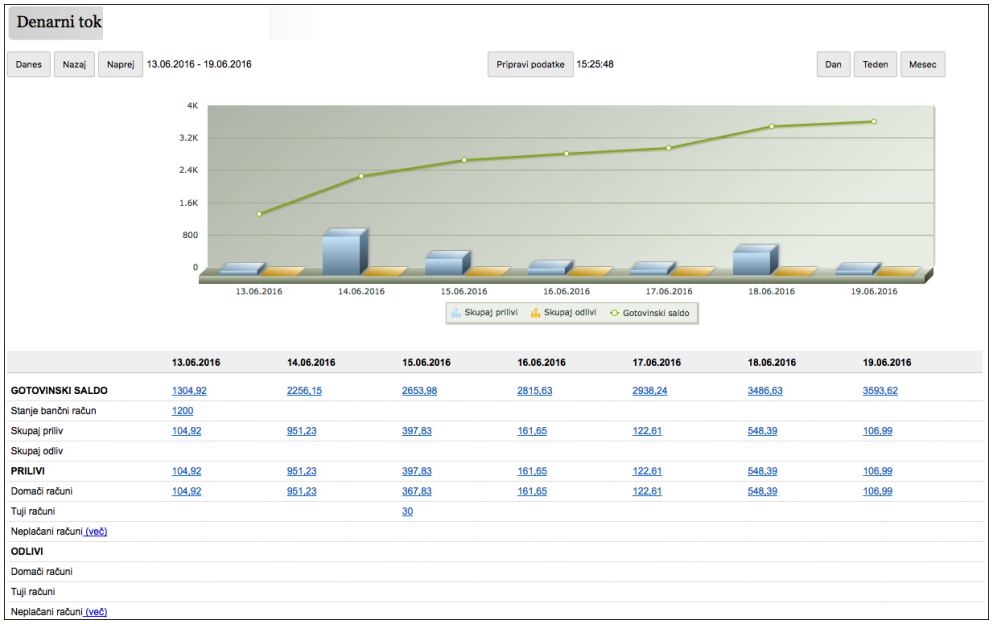
Aplikacija ne predvideva zajem podatkov iz plačilnih inštrumentov, zato je potrebno na dan pogleda denarnega toka podatke ročno vpisati v aplikacijo. Kot kaže slika 4.9, izberemo polje ob plačilnem inštrumentu in vpišemo vrednost. Ob shranitvi podatkov se izvede ponovno preračun denarne napovedi.

Slika 4.9: Ročni vnos stanja na plačilnih inštrumentih

4.3.4 Navigacija po podatkih

Glavni del uporabniškega vmesnika aplikacije za denarni tok je razdeljen na dva dela :

- graf prihodkov, odhodkov in trenutnega stanja,
- tabela s podatki v kategorijah prihodkov in odhodkov,



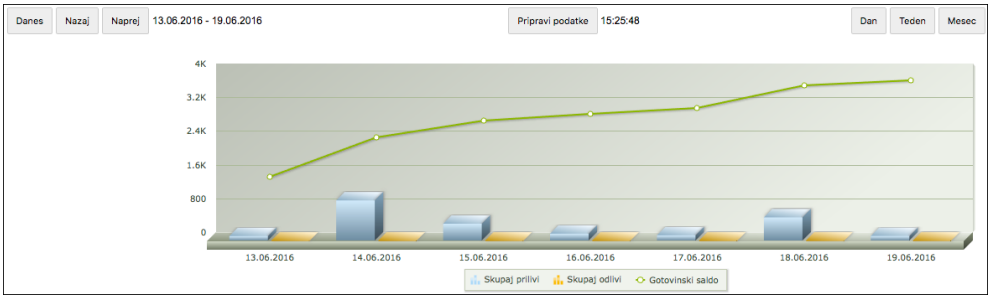
	13.06.2016	14.06.2016	15.06.2016	16.06.2016	17.06.2016	18.06.2016	19.06.2016
GOTOVINSKI SALDO	1304,92	2256,15	2653,98	2815,63	2938,24	3486,63	3593,62
Stanje bančni račun	1200						
Skupaj priliv	104,92	951,23	397,83	161,65	122,61	548,39	106,99
Skupaj odliv							
PRILIVI	104,92	951,23	397,83	161,65	122,61	548,39	106,99
Domači računi	104,92	951,23	367,83	161,65	122,61	548,39	106,99
Tuji računi			30				
Neplačani računi (več)							
ODLIVI							
Domači računi							
Tuji računi							
Neplačani računi (več)							

Slika 4.10: Uporabniški vmesnik

Za navigacijo po podatkih smo implementirali podobno uporabniško izkušnjo, kot jo imajo aplikacije koledarji (glej 4.10):

- levo zgoraj se nahajajo gumbi za premik po časovni osi naprej in nazaj ter vrnitev na današnji dan
- "Pripravi podatke" ponovno prebere vse podatke ter izračuna napovedi plačil
- Dan / Teden / Mesec predstavljajo časovno komponento po kateri gledamo skupaj združene prihodke / odhodke v eni koloni tabele oz. stolpca grafa.

Na grafu (glej 4.11) pa so prikazani skupni prilivi in odlivi ter črta, ki nam prikazuje gotovinski saldo.



Slika 4.11: Prikaz grafa

Tabelarični prikaz podatkov je namenjen prikazu računov po kategorijah in možnost kopanja (ang. drilldown) po podatkih. Na levi strani (glej 4.12) se nahajajo kategorije :

- Odlivi so vhodni računi, ločeni na domače in tuje račune (določimo v Excel tabeli pri uvozu). Neplačani računi so skupina računov, ki še niso plačani in imajo rok plačila v preteklosti.
- Prilivi so izhodni računi. Ostala logika je enakovredna logiki odlivov
- Gotovinski saldo je končni seštevek denarja, ki ga sestavljajo seštevki po plačilnih inštrumentih (npr. stanje bančni račun), dodano Skupaj prilivi in odšteto Skupaj odlivi.

	13.06.2016	14.06.2016	15.06.2016	16.06.2016	17.06.2016	18.06.2016	19.06.2016
GOTOVINSKI SALDO	1304,92	2256,15	2653,98	2815,63	2938,24	3486,63	3593,62
Stanje bančni račun	1200						
Skupaj priliv	104,92	951,23	397,83	161,65	122,61	548,39	106,99
Skupaj odliv							
PRILIVI	104,92	951,23	397,83	161,65	122,61	548,39	106,99
Domači računi	104,92	951,23	367,83	161,65	122,61	548,39	106,99
Tuji računi			30				
Neplačani računi (več)							

Slika 4.12: Tabelarični prikaz podatkov

Za vsako številko lahko preverimo, na podlagi katerih računov je bila izračunana. Izpiše se seznam računov z osnovnimi podatki. Ob kliku na

napoved plačila lahko izvemo, kako je bila določena napoved glede na rok plačila.

Račun	Partner	Znesek	Datum	Rok plačila	Napoved plačila
573/ 2016		38,43	03.05.2016	13.05.2016	14.06.2016 Zadnje 3 računi (179/ 2016, 26 / 2016, 1191/ 2015) so imeli povprečno 32 dni odstopanja od roka plačila
683/ 2016		30,74	30.05.2016	07.06.2016	14.06.2016
705/ 2016		18,3	03.06.2016	17.06.2016	14.06.2016

Slika 4.13: Prikaz pojasnila za napoved plačila

4.3.5 Neplačani računi

V napovedi denarnega toka gledamo podatke v prihodnost od datuma gledanja naprej. Po tem kriteriju so vključeni tudi računi, ki še niso bili plačani, rok plačila je tudi v preteklosti, vendar je napoved prejetja plačila v prihodnosti. Na drugi strani pa ostanejo izven prikaza računi, katerim je rok plačila že zapadel in je tudi napoved prejetja plačila v preteklosti. Le ti se nahajajo v skupini Neplačani računi, za katere lahko določimo :

- Ocena plačila - datum, ko predvidevamo, da bo račun plačan. S tem račun postane del napovedi denarnega toka,
- Neizterljivo - račun ne bo nikoli plačan (npr. podjetje je šlo v stečaj in ni pričakovati, da bomo dobili povrnjen znesek iz stečajne mase), zato je najbolje, da prihodek odpišemo.

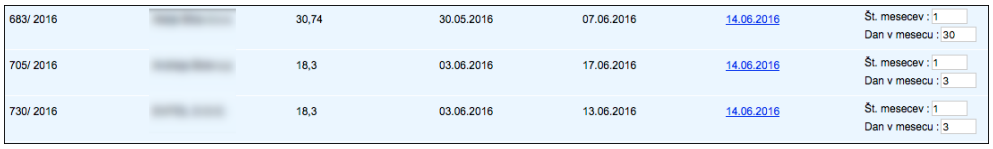
Neplačani računi (več)

Račun	Partner	Znesek	Datum	Rok plačila	Napoved plačila
15/ 2013		19,44	03.01.2013	16.01.2013	04.03.2014
92/ 2013		19,44	04.02.2013	17.02.2013	05.04.2014
143/ 2013		19,44	03.03.2013	16.03.2013	02.05.2014
196/ 2013		2,88	02.04.2013	09.04.2013	17.06.2013

Slika 4.14: Vpis ocene prejema ali označba Neizterljivo na neplačanih računih

4.3.6 Vnos periodično izdanih računov

Za vsak račun lahko vnesemo obdobje ponavljanja, ki se potem upošteva pri napovedih za prihodnost. Tako lahko simuliramo znane prihodne in odhodke, ki se bodo pojavili v prihodnosti.



Slika 4.15: Vnos periodično izdanih računov

4.4 Primer uporabe aplikacije

Potek celotnega dela aplikacije bomo predstavili na enostavnem primeru (glej 4.16). Podatke bomo gledali na dan 1.6.2016, v podjetju pa načrtujemo naslednje prihodke :

- enkratna priliv od podjetja Mercator na dan 2.6.2016,
- ponavljajoči mesečni priliv od podjetja Krka vsakega 3. v mesecu,
- zapadel račun od podjetja Sava, ki bi že moral biti plačan.

Na odhodkovni strani imamo planirane naslednje mesečne izdatke :

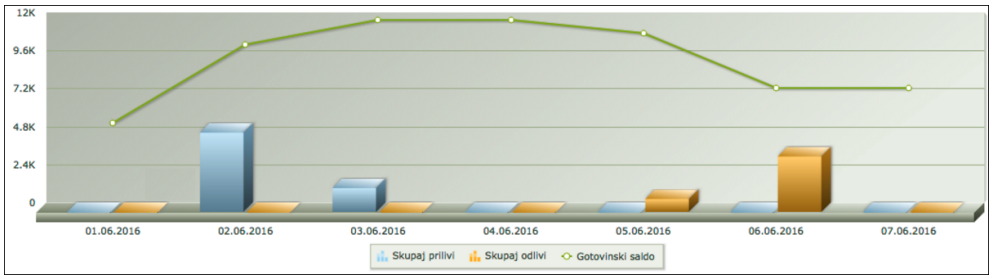
- najem poslovnih prostorov, ki jih moramo plačati vsakega 5. v mesecu,
- plače zaposlenim vsakega 6. v mesecu.

Dne 1.6.2016 imamo tudi na bančnem računu stanje 5000 EUR.

Področje	Tip	Številka računa	Partner	Neto znesek	Datum	Rok plačila	Datum plačila
Priliv	Domači	D101	Mercator d.d.	5000	25.5.2016	2.6.2016	
Priliv	Domači	D102	Krka d.d.	1500	26.5.2016	3.6.2016	
Priliv	Domači	D103	Sava d.d.	4000	1.5.2016	15.5.2016	
Odliv	Domači	P100	Poslovni prostori d.o.o	800	27.5.2016	5.6.2016	
Odliv	Domači	Plače	Moje podjetje	3500	21.6.2016	6.6.2016	

Slika 4.16: Vhodni podatki za primer uporabe aplikacije

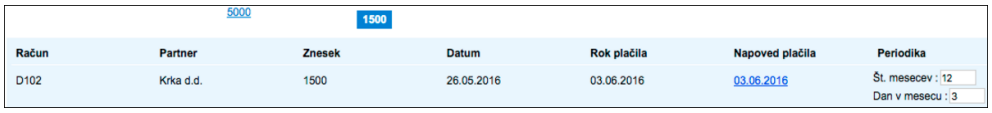
Ob uvozu podatkov dobimo graf, kot je prikazan na sliki 4.17.



Slika 4.17: Graf po uvozu začetnih podatkov

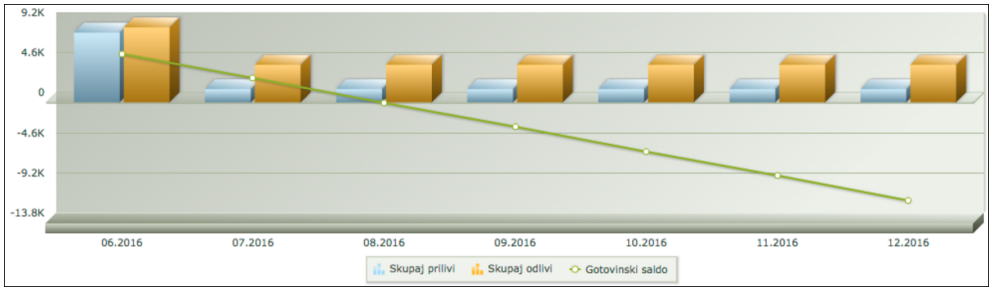
Iz hitrega pogleda lahko sklepamo, da je poslovanje dobro - na začetku meseca smo imeli dobre prilive, potem nekaj odlivov, ampak stanje po tem je še vedno boljše kot na začetku meseca. Vendar videz vara, ker gledamo stanje samo za nekaj dni.

Za bolj natančno analizo naprej vnesemo podatke o periodično izdanih računih, kot prikazuje spodnja slika.



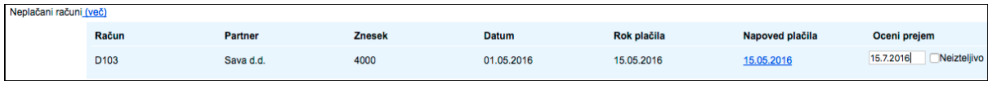
Slika 4.18: Vpis podatkov za periodične račune

Preko navigacije po podatkih si ogledamo mesečni pogled na stanje denarnega toka. Iz slike 4.18 razberemo, da smo 6. meseca imeli sicer dobre prihodke, vendar so na mesečni ravni še vedno dosti nižji, kot pa stroški. Zato nam bo 9. meseca že zmanjkalo denarja za poslovanje.



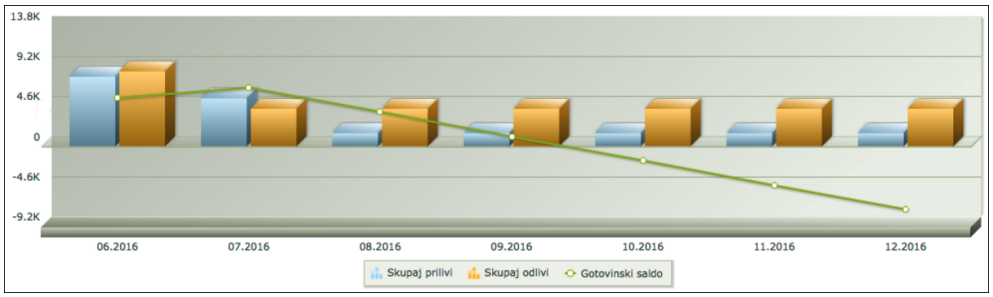
Slika 4.19: Graf po določitvi periodičnih računov

Situacijo lahko izboljšamo, če se nam uspe dogovoriti z kupcem, pri katerem imamo zapadel račun. Za primer predpostavimo, da smo se za plačilo dogovorili na datum 15.7. Podatek lahko vnesemo pod zapadle račune, kot prikazuje spodnja slika.



Slika 4.20: Vpis ocene plačila na neplačan račun

Ob ponovno pogledu mesečnega grafa lahko ugotovimo, da se bo naš denarni tok nekoliko popravil, vendar nam bo na daljši rok še vedno zmanjkalo denarja. Če želimo svoje obveznosti redno poravnati, moramo pridobiti posel, ki nam bo popravil našo finančno sliko prihodkov.



Slika 4.21: Končni graf po vpisu ocene plačila na neplačan račun

Poglavje 5

Sklepne ugotovitve

V diplomski nalogi smo razvili enostavno aplikacijo za spremljanje denarnega toka podjetja. Naloga uporabnika je, da si ključne podatke vhodnih in izhodnih računov pripravi v obliki Excel datoteke in jih uvozi v sistem. Vpiše še trenutno stanje denarja na različnih plačilnih inštrumentih in že pridobi zelo podrobne vpogled, kako se bo stanje njegovega denarja gibalo v naslednjih tednih in mesecih glede na predvidene prihodke in odhodke. Sprehod po podatkih je prilagojen uporabniški izkušnji spletnih koledarjev, kar poenostavi preglednost podatkov. Vsak sešteti podatek (npr. prihodki za določen mesec) omogoča kopanje v globino (ang. *drilldown*) do izvora podatkov (računa). Enostaven algoritem za napoved roka prilivov in odlivov omogoča jasen vpogled v postopek pridobivanja rezultata in posledično nudi uporabniku večjo stopnjo zaupanja v napoved sistema. Eden od večjih izzivov pri razvoju aplikacije je bila omejitev funkcionalnosti na nabor najbolj osnovnih in uporabnih funkcionalnosti. Pojavilo se nam namreč je veliko idej, kako bi sistem naredili uporaben v različnih robnih primerih. Vendar se tak pristop v praksi običajno izkaže kot napačen, saj pripelje do prevelike kompleksnosti rešitve in uporabe le te za običajne uporabnike.

5.1 Možne nadgradnje obstoječega sistema

Med razvojem sistema smo se srečali s kar nekaj večjimi idejami, kako izboljšati točnost napovedi. Večina predlogov je povezana z izboljšanjem kvalitete informacij, ki v prihodnosti pripomorejo k napovedi izdaje računa in posledično njihovem plačilu.

5.1.1 Akontacija davka

Davek od dohodkov pravnih oseb se plačuje na podlagi letnega davčnega obračuna. V tem obdobju podjetje pridobi tudi znesek za letno akontacijo, ki ga plačuje na podlagi mesečnega ali trimesečnega obdobja. V primeru normalnega poslovanja podjetja je to kar znesek fiksnega mesečnega stroška preko celotnega leta, zato bi ga aplikacija lahko avtomatično upoštevala. Na podlagi predvsem zgodovine vhodnih in izhodnih računov bi lahko podali dobro oceno za resnično razliko davka na dodano vrednost glede na plačano akontacijo. Ta podatek bi bil osnova za oceno, ali bomo morali po oddaji bilance plačati še dodatni davek ali bomo dobili določen znesek povrnjen.

5.1.2 Popusti v primeru plačil pred rokom

Večina podjetij ima zelo malo možnosti vpliva na plačilo obveznosti, če so njihove storitve ali izdelki že v celoti dostavljene kupcu. V tem primeru lahko računa samo na poštenost kupca in na njegov interes dolgoročnega sodelovanja. Ostali postopki (izterjava, sodni postopki) običajno trajalo veliko časa in zahtevajo dosti vložka denarja in časa. Zato se v praksi velikokrat uporabi možnost dodatnega popusta v primeru predplačila blaga ali storitve (npr. kupec pridobi dodatnih 5Zelo zanimiva razširitev bi bila možnost, da za izdane račune lahko simuliraš plačilo pred rokom. Tako bi lahko v aplikaciji nastavil, da imajo vsi izdani računi 5

5.1.3 Ponudbe

Ponudbe se v poslovnem svetu običajno uporabljajo za informativno povpraševanje o ceni, lahko pa so že potrditev dogovorjene opravljene storitve ali dobave blaga. V tem primeru so zelo uporabna informacija za denarni tok, saj nam dajo veliko zagotovilo, da bo storitev ali blago tudi plačano v predvidene roku. Kar še posebej velja za ciklične dobave, ko večkrat na leto ali mesec istemu kupcu dobavljano enake ali podobne storitve oz. izdelke. Napoved denarnega toka bi lahko iz množice ponudb izluščila ponudbe, ki ustrezajo zgornjim kriterijem ter posledično upoštevala, da bo za njih v bližnji prihodnosti tudi izdan račun.

5.1.4 Delovni nalogi

Pri podjetjih, kjer glavni vir prihodkov predstavljajo storitve ali izdelki, ki jih sami izdelajo v daljšem obdobju, običajno njihovo proizvodnjo spremljajo preko delovnih nalogov. V tem primeru je izdaja računa in posledično prihodek zelo odvisen od tega, kako hitro je podjetje sposobno zaključiti predvidene izdelke. Podobno kot pri ponudbah se tudi tukaj v praksi izkaže, da je veliko dobav enim in istim kupcem. Napoved denarnega toka bi tako preko preteklih računov identificirala delovne naloge, ki so se v preteklosti že ponavljali. Običajno delovni nalog sestavlja tudi popis opravljenega dela, ki vsebuje ure in datuma. Na podlagi tega se lahko ugotovi koliko koledarskega in absolutnega časa je bilo porabljeno za izdelavo ene enote mere izdelka. Ta podatek pa nam lahko koristi kot ocena za čas izdelave planiranih ali začetnih delovnih nalogov in posledično za čas zaključitve in izdaje računa.

5.1.5 Mobilni vpogled v podatke

Razširjenost mobilnih telefonov v zadnjih letih zahteva tudi dostop do vsaj osnovnih informacij na le teh napravah. Zato menimo, da bi bila zelo zanimiva razširitev prikaz in sprehajanje po podatkih na mobilnih napravah.

5.1.6 Črpanje podatkov iz različnih virov

Trenutna rešitev predvideva, da se podatki preberejo preko Excel datoteke. Pristop je zamuden, če moramo iz obstoječega informacijskega sistema podatke naprej pretvoriti v Excel obliko, jih potem še dodatno obdelati po predpisanih pravilih in na koncu uvoziti v aplikacijo. Glede na to, da kar nekaj poslovnih programov v Sloveniji že podpira dostop do podatkov preko programskega vmesnika, bi lahko aplikacija tudi omogočala priklop na vmesnik in zatem neposredno branje živih poslovnih podatkov.

5.2 Spremna misel

Napoved denarnega toka sicer omogoča napovedovanje do dneva natančno, vendar se v praksi izkaže, da take meritve niso najbolj merodajne. Podjetja ne moremo poslovati iz dneva v dan, ker obstaja kar nekaj zakonskih obveznosti (plačilo davka, plače, itd), ki jih moramo redno plačati in posledično moramo imeti določen del tekočega denarja tudi na zalogi. Veliko bolj so smiselne tedenske napovedi, saj zajamejo tudi manjša odstopanja, ki so povezana z nenačrtnim zamujanjem pri plačilih. Bistveno bolj je zanimivo spremljanje in napoved denarnega tako iz strani prodaje, kot pa denarni tok nabave. Nabavni tok lahko namreč v večji meri urejamo sami z odločitvami, kako bomo plačevali svoje obveznosti. Seveda pa lahko izstopajo večji nepredvidljivi dogodki (okvara, odtujitev oz. kraja opreme, itd). in tako se večina podjetji odloči, da svoje obveznosti poplača na rok plačila oz. imajo konstantne roke zamujanja. Prototip aplikacije, ki smo jo predstavili v diplomski nalogi, je predvsem primeren za mala in srednja podjetja. V teh velja, da je vedno premalo časa za administrativno spremljanje poslovanja podjetja, saj se je potrebno ukvarjati s pridobivanjem poslov, izvedbo del, plačili in z izterjavami. Zato je vsako orodje, ki jim poenostavi vpogled v poslovanje, zelo dobrodošlo. Pri velikih podjetjih pa se srečamo z dosti bolj zahtevnimi poslovnimi procesi. Rešitve za napovedovanja denarnega toka gredo običajno v dve smeri. Srečamo sisteme, ki poizkušajo zajeti prav vse informacije v

podjetju oz. spremljajo vsak izhod poslovnega procesa. Na drugi strani pa so enostavna pravila, ki lastnikom oz. upravi omogočajo spremljanje delovanja poslovalnic. Npr. v enem večjem slovenskem trgovskem podjetju, ki je del velike mednarodne verige trgovskih podjetji, imamo zelo enostaven sistem spremljanje denarnega toka : čisto vsak nakup mora biti opravljen izključno iz tekočega denarnega toka (brez kreditov) in na tekočem računu podjetja mora biti vedno vsota denarja, ki podjetju omogoča preživetje 3 mesece brez prihodkov. Predvsem prvo pravilo - nobenih investicij na zadolževanje - lastnikom omogočajo resnično enostavno kontrolo, da podjetje ne zaide v težave. Verjetno je tak pristop primeren tudi za kakšno manjše ali slednje podjetje.

Literatura

[1] Mramor Dušan. "Uvod v poslovne finance", Gospodarski vestnik, 1993, str. 381

[2] NLB, "Denarni tok odloča o usodi podjetja" [Online]. Dosegljivo: https://www.nlb.si/denarni-tok-odloca-o-usodi-podjetja, 2014 [Dostopano 28.5.2016]

[3] Franci Zupan : Kreditiranje majhnih in srednje velikih podjetij v finančnem posredništvu [Online]. Dosegljivo: http://www.cek.ef.uni-lj.si/u_diplome/zupan620.pdf. Ljubljana, 2003, str 28

[4] openIT, "Denarni tok je krvni obtok podjetja - poskrbite zanj" [Online]. Dosegljivo: http://openit.si/izobrazevanje/denarni-tok-je-krvni-obtok-podjetja-poskrbite-zanj, 2016 [Dostopano 28.5.2016]

[5] GWT project, "JSNI" [Online]. Dosegljivo: http://www.gwtproject.org/doc/latest/DevGuideCodingBasicsJSNI.html, 2016 [Dostopano 28.5.2016]

[6] GWT project, "JsInterop v1.0: Nextgen GWT/JavaScript Interoperability" [Online]. Dosegljivo: https://docs.google.com/document/d/10fmlEYIHcyead_4R1S5wKGs1t2I7Fnp_PaNaa7X
2015 [Dostopano 28.5.2016]

[7] Adam Tacy, Robert Hanson, Jason Essington, and Anna Tökke, "GWT in Action, Second Edition", Manning publications, 2013, str. 196

[8] Wikipedia, "HTML" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/HTML, 2016 [Dostopano 28.5.2016]

[9] Wikipedia, "CSS" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Cascading_Style_Sheets, 2016 [Dostopano 28.5.2016]

[10] Joda-Time, "Quick start guide" [Online]. Dosegljivo: http://www.joda.org/joda-time/quickstart.html, 2016 [Dostopano 28.5.2016]

[11] Apache POI, "Busy Developers' Guide to HSSF and XSSF Features Busy Developers' Guide to Features" [Online]. Dosegljivo: https://poi.apache.org/spreadsheet/quick-guide.html, 2016 [Dostopano 28.5.2016]

[12] FusionCharts, "FusionCharts Developer Center" [Online]. Dosegljivo: http://www.fusioncharts.com/dev/, 2016 [Dostopano 28.5.2016]

[13] Wikipedia, "Scrum (software development)" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Scrum_(software_development), 2016 [Dostopano 28.5.2016]

[14] Atlassian, "A brief introduction to kanban" [Online]. Dosegljivo: https://www.atlassian.com/agile/kanban, 2016 [Dostopano 28.5.2016]